**Hyperledger Fabric**

# Concept

* Private and Permissioned Blockchain
* Hyperledger Fabric은 네트워크 참여자를 믿을 수 있는 Membership Service Provider(MSP)를 통해 참여시킬 수 있다.
* 몇몇의 Pluggable options
  + Ledger data를 다양한 format으로 저장할 수 있다.
  + Consensus mechanisms를 변경할 수 있다.
  + Different MSPs를 지원한다.
* “Channels”를 만들 수 있다.
  + 별도의 참가자 그룹이 개별의 트랜잭션 장부를 만드는 것을 허용한다.
  + 이는 privacy transaction, data를 위해 매우 유용하다.
* Ledger
  + World state
    - Given point 시점에서의 장부의 상태를 표현한다.
    - Database of the ledger
  + Transaction log
    - 현재 world state가 되기 까지의 모든 트랜잭션을 기록한다.
    - Simply recoeds the before and after values of the ledger database
    - Update history for the world state
  + Ledger은 world state를 대체할 수 있는 데이터 저장소를 가지고 있다.
    - Default : LevelDB (Key – value ) store database
* Smart Contract
  + Chaincde로 작성된다.
  + 어플리케이션이 Ledger와 상호작용해야 할 경우 어플리케이션에 의해 호출된다.
  + 대부분의 경우, chaincode는 world state와만 querying it 과 같은 형태로 상호작용한다.( not the transaction log )
  + 현재 Go, Node.js 언어를 지원한다.

# Smart Constracts [ 기존 블록체인 플랫폼 ]

* 블록체인 플랫폼 스마트 컨트랙트 대부분은 order-execute 구조의 합의 알고리즘을 사용한다.

1. 검증되고 요청된 트랜잭션은 모든 피어 노드에 전파된다.
2. 각 피어 노드는 트랜잭션을 순차적으로 실행한다.

* order-execute 구조의 블록체인에서 스마트 컨트랙트의 실행은 반드시 완결성을 띄어야한다. 그렇지 않으면 합의에 도달할 수 없다. -> 비결정적인 실행은 제거된다 -> 이것은 광범위한 선택을 방해한다.(언어적인 선택)
* 모든 트랜잭션은 모든 노드에서 순차적으로 실행되기 때문에 Performance와 scale이 제한적이다.
* 스마트 컨트랙트 코드가 시스템 모든 노드에서 실행된다는 사실은 복잡한 측정을 요구한다. 이는 잠재적으로 발생할 수 있는 악의적인 코드로 부터 시스템 전체를 보호하고 시스템 전체를 탄력성을 보장하기 위해서 이다.

# Smart Contracts [ Hyperledger Fabric ]

* execute-order-validate architecture
  + address : resiliency, flexibility, scalability
  + 성능과 기밀 유지 문제 : 아래 3번의 단계를 통해 트랜잭션을 분리하여 order-execute 모델에 대응한다.

1. execute : 트랜잭션을 실행하고 그 정확성을 확인함으로써 해당 트랜잭션을 보증한다.
2. Order : 합의 알고리즘을 통해 트랜잭션 요청
3. Validate : 원장에 commit 되기 전에 어플리케이션별 endorsement policy에 대해 트랜잭션의 유효성을 검증한다.

* 이는 합의 전에 트랜잭션이 실행된다는 점에서 기존의 order-execute 방식과는 크게 다르다.
* 어플리케이션별 endorsement policy는 해당 스마트 컨트랙트의 옳바른 실행을 보증하기 위해 어떤 노드 혹은 얼마나 많은 노드가 보증해야하는지를 지정한다.
  + 각 트랜잭션은 해당 트랜잭션의 endorsement policy에 충족하는데 필요한 피어 노드의 하위 집합에서만 실행되면 된다. => 동시 실행을 허용하기 때문에 성능과 확장성의 증가
  + Non-determinism을 제거하였기 때문에 Fabric은 표준 프로그래밍 언어(Go, Node.js 등)를 사용할 수 있는 첫번째 블록체인 기술이다.

# Privacy and Confidentiality

* 기존 블록체인은 모든 노드에서 트랜잭션이 실행되어야 했기 때문에 컨트랙트와 해당 컨트랙트 실행을 위한 트랜잭션 데이터 또한 모든 노드가 가지고 있다. 이 때문에 네트워크의 모든 노드는 컨트랙트 코드, 트랜잭션 정보를 볼 수 있다. => 기밀성이 보장되지 않는다.
* **Public Blockchain**
  + **암호화된 데이터** : 암호화된 데이터는 기밀성을 제공하기 위한 하나의 접근법이다. 하지만 Permissionless Blockchain의 경우 모든 노드가 암호화된 데이터를 가지고 있다. 이는 암호화된 데이터를 복호화 하기에 충분한 시간과 재료를 재공하는 것이기 때문에 암호화는 유지될 수 없다.
  + **Zero Knowledge Proofs(ZKP)** : ZKP의 경우 계산하는데 상당한 시간과 계산 리소스가 필요하기 때문에 기밀성을 보장하기 위한 절충안으로 평가되고 있다.
* **Hyperledger Fabric**
  + Channel architecture을 통해 기밀성을 보장할 수 있다.
  + Fabric Network 참여자는 기본적으로 하위 그룹 사이의 채널을 형성할 수 있다. 채널은 해당 채널의 참가자들에게 트랜잭션 그룹의 visibility를 보장한다.
  + 따라서, 해당 채널에 참가자 노드만이 smart contract와 data transacted에 접근할 수 있다.
  + Fabric은 v1.2이후에 ZKP를 통한 private data 지원한다. -> private data collection을 통해

# Pluggable Consensus

* Hyperledger Fabric은 수정 가능한 consensus protocol을 지원한다.
* 현재 Fabric에서 제공하는 CFT ordering service는 Kafka와 Zookeeper를 통해 구현되었다.
* 이후에 Fabric은 Raft와 fully decentralized BFT ordering service를 통해 구현된 Raft consensus ordering service를 지원 예정이다.
* Fabric network는 다양한 ordering service를 가질 수 있지만 하나의 어플리케이션에서는 하나의 ordering service만 가능하다.